Studium wykonalności

3@KASK

7 kwietnia 2009

Symbol projektu: 3@KASK	Opiekun projektu: mgr inż. Tomasz Boiński
Nazwa Projektu: Wizualizacja grafów za pomocą biblioteki Prefuse	

Nazwa Dokumentu: Studium wykonalności	Nr wersji: 0.4
Odpowiedzialny za dokument: Anna Jaworska	Data pierwszego sporządzenia: 31.03.09
Przeznaczenie: WEWNĘTRZNE	Data ostatniej aktualizacji: 31.03.09

Historia dokumentu

Wersja	**Opis modyfikacji**	**Rozdział/strona**	**Autor modyfikacji**	**Data**
0.0	Przygotowanie zarysu dokumnetu i określenie zakresu badań	wszystkie	Anna Jaworska	31.03.09
0.1	Zdefiniowanie wymagań	3	Cały zespół	31.03.09
0.2	Dołaczenie opisu poprawnego tworzenia bibliotek	3.5	Radosław Kleczkowski	01.04.09
0.3	Dołączenie opisów bibliotek graficznych	6.2	Piotr Kunowski	02.04.09
0.4	Opis uwarunkowań prawnych i rozszerzenie opisu wariantów	6.1	Anna Jaworska	06.04.09
0.5	Uzupełnienie braków	wszystkie	Cały zespół	07.04.09
0.6	Dołączenie opisu odmian języka OWL	głównie 6,7	Piotr Orłowski	07.04.09

Spis treści

1 Założenia realizacji studium

1.1 Podstawa wykonania i temat studium

Studium wykonywane jest przede wszystkim aby określic możliwe sposoby realizacji projektu. Ma także za zadanie zebranie i podsumowanie informacji potrzebnych zespołowi do realizacji projektu.

1.2 Cel studium

Celem studium jest zbadanie na potrzeby projektu *Wizualizacja grafów za pomocą biblioteki Prefuse*:

- jak należy tworzyć biblioteki w technologii JAVA
- jakich mechnizmów wizualizacji grafów dostarczają biblioteki JAVA
- czy realizacja projektu za pomocą Prefuse jest odpowiednim rozwiązaniem
- jaki standard OWL powinien być wspierany przez wytworzony produkt

1.3 Ograniczenia

Do podstawowych ograniczeń należą:

- konieczność realizacji projektu w języku JAVA
- konieczność wykorzystania wersji bibliotek zgodnych z użytymi w OCS
- limit czasowy projektu

2 Stan istniejący

2.1 Inne systemy i zasoby mające wpływ lub będące pod wypływem planowanego produktu

- OCS - Ontology Creation System
- OWL API ver 2.1.1 - API do przetwarzania plików w formacie OWL zgodnych ze standardem W3C; ta wersja API została użyta w projekcie OCS
- biblioteki graficzne - w szczególności Prefuse

2.2 Istniejące na rynku podobne rozwiązania

- Protege - bardzo znany system do edycji i wizualizacji ontologii autorstwa Stanford University. Napisany w JAVA. Ze względu na fakt, iż jest aplikacją standalone, wykorzystującą stosunkowo duże zasby systemowe i tudną do integracji z portalem OCS, nie może zostać wykorzystana jako gotowe rozwiązanie.

2.3 Problem i motywacja wdrożenia nowego produktu

Nowa biblioteka powinna powstać aby:

- ułatwić programistom wizualizację ontologii
- zapewnić API pozwalające na bezpośrednią translację OWL na postać graficzną
- zapewnić rozwiązane przenośnie i uniwersalne

3 Ogólne wymagania stawiane produktowi i ich priorytety

Wymienione wymagania mają charakter orientacyjny, pozwalający nakreślić zakres problemu jaki ma pokrywać projekt. Szczegółową definicję wymagań będzie zawierać dokument *Specyfikacji wymagań*. W szczególność możliwe jest, że niektóre z wymienionych poniżej wymagań zostaną usunięte lub zmieniona oraz mogą zostać dodane inne wymagania.

3.1 Użytkownicy

Użytkownikami biblioteki będą programiści tworzący aplikacje wizualizujące ontologie. Inicjalnie będą to programiści związani z projektem OCS, później mogą to być dowolni inni programiści chętni do korzystania z biblioteki.

3.2 Dane

Obsługiwane formaty Biblioteka powinna obsługiwać te same formaty danych co OWL API (zgodne ze specyfikacją W3C):

- RDF
- RDF Schema
- OWL Lite
- OWL DL
- OWL Full

Wczytywanie danych Ponadto dane te powinny być wczytywane poprzez:

- podanie ścieżki do pliku OWL na dysku
- podanie adresu sieciowego zasobu z plikiem OWL
- podanie strumienia/kontenera z XML

Modyfikowalność danych Biblioteka powinna udostępniać metody do modyfikacji wczytanych danych i możliwość zapisania zmienionych danych. Dane powinny być dostarczane użytkownikowi w postaci strumienia/kontenera z poprawnie sformatowanym plikiem OWL. Biblioteka nie musi sprawdzać czy zmiany wprowadzone przez użytkownika są logicznie poprawne.

3.3 Funkcjonalność

Zakładamy, że biblioteka będzie zawierać następujące funkcjonalności:

- wizualizacja elmentów OWL
- pozwalać użytkownikowi na definiowanie akcji dla zdarzeń okna
- zawierać standardowe definicje zdarzeń
- wczytywanie, modyfikowanie i zapis ontologi
- definiowanie parametrów wyglądu, w szczególności ilości widocznych poziomów grafu

3.4 Wymogi techniczno - technologiczne

3.4.1 Standard tworzenia biblioteki

Nie istnieją żadne formalne zalecenia dotyczące tworzenia bibliotek JAVA. Są jednak pewne zalecenia co do stosowanych praktyk [**?**]:

1. **Odpowiednie kapsułkowanie.** Publiczne powinny być jedynie te klasy i metody, które są istotne dla użytkownika i z których będzie on bezpośrednio korzystał.
2. **Możliwość debugowania.** Użytkownik powinien mieć możliwość debugowania kodu biblioteki, bez konieczności znajomości każdego jej szczegółu.
3. **Przejrzystość.** Kod biblioteki powinien być odpowiednio udokumentowany za pomocą javadoc. W szczególności, bardzo dokładnie należy opisać klasy oraz metody publiczne.
4. **Łatwość użycia.** Biblioteka powinna zawierać klasy, pokazujące przykłady wykorzystania jej klas i metod.
5. **Rozszerzalność.** Struktura wewnętrzna biblioteki powinna być odpowiednio podzielona na klasy (wykorzystując klasy abstrakcyjne i interfejsy. Dzięki temu użytkownik będzie miał możliwość stworzenia własnych klas, rozszerzających funkcjonalność biblioteki.
6. **Uniwersalność.** Biblioteka powinna mieć jasno określony problem, który rozwiązuje. Wyniki powinny być podane użytkownikowi w wygodny dla niego sposób (lub na kilka sposobów), który będzie umożliwiał wykorzystanie biblioteki w różnych aplikacjach. Innymi słowy, biblioteka powinna udostępniać łatwy i przejrzysty dla użytkownika interfejs.
7. Biblioteka powinna być napisana w taki sposób, aby użytkownik spojrzał na nią i mógł powiedzieć: "Wow, to jest dokładnie to, czego potrzebuję i dokładnie tak samo bym to napisał!" ;).

4 Ogólna ocena ryzyka i planowany sposób zarządzania nim

Schemat opisu czynnika ryzyka

ID czynnika	RISKXX
Nazwa czynnika	Nazwa
Opis czynnika	Opis...
Sposób zarządzania	Opis..

4.1 Czynniki ryzyka

ID czynnika	RISK01
Nazwa czynnika	Problemy logistyczne zespołu
Opis czynnika	Uwzględniamy możliwość wystąpienia problemów osobistych członków zespołu powodujących ich wyłączenie z prac.
Sposób zarządzania	Jeśli ktoś zostanie wyłączony z prac, reszta zespołu musi podzielić między siebie jego obowiązki i informować osobę wyłączoną o postępach, tak aby miała wgląd w postęp prac, które miała wykonywać i kontynuować je po niedyspozycji.

ID czynnika	RISK02
Nazwa czynnika	Problemy członków zespołu na uczelni
Opis czynnika	Możliwe jest powstanie zaległości związanych z innymi uczelnianymi obowiązkami
Sposób zarządzania	Członek zespołu musi zgłosić swoje problemy reszcie zespołu. W zależności od sytuacji termin wykonania jego zadań zostanie przedłużony lub zadania te przejmie ktoś inny.

ID czynnika	RISK03
Nazwa czynnika	Niedostępność opiekuna/klienta
Opis czynnika	Z różnych przyczyn niezależnych od zespołu opiekun może stać się niedostępny.
Sposób zarządzania	Wszelkie problemy wymagające, według zespołu, poznania opinii opiekuna będą musiały zostać rozwiązanie poprzez podjęcie decyzji przez zespół bez wsparcia. Wszelkie problemy organizacyjne związane z projektem grupowym powinny być pod nieobecność opiekuna zgłaszane do katedralnego koordynatora projektów grupowych.

ID czynnika	RISK04
Nazwa czynnika	Niewystarczająca wiedza programisty
Opis czynnika	W trakcie pisania kodu może okazać się, że programista z powodu nieznajomości bibliotek/metod/praktyk zacznie mieć problemy z wydajnym kodowaniem (zacznie popełniać częste błędy, pracować bardzo wolno).
Sposób zarządzania	Osoba mająca problemy z danym kodem powinna zgłosić to reszcie zespołu. Jeśli ograniczenia czasowe na to pozwolą dostanie ona dodatkowy czas na wykonanie zadania. Jeśli nie będzie to możliwe, zadanie zostanie przekazanie osobie będącej w stanie poradzić sobie z zagadnieniem lub zostanie podzielone między większą liczbę osób.

ID czynnika	RISK05
Nazwa czynnika	Awaria SVN
Opis czynnika	Serwer SVN nie jest dostępny lub działa w sposób nieporządany.
Sposób zarządzania	Problem należy niezwłocznie zgłosić opiekunowi i oczekiwac na jego interwencję.

5 Uwarunkowania prawne i inne

Docelowy produkt będzie własnością KASK. Należy zadbać o to aby używane w projekcie biblioteki były na licencjach pozwalających na użycie w produkcie zamkniętym. W szczególności należy zwrocić uwagę aby biblioteki nie były na licencji GPL, która narzuca produktowi również licencję GPL, co sprawia, że produkt nie musi zostać upubliczniony.

6 Proponowane rozwiązania

Rozważane rozwiązania zostaną rozważone pod względem wersji OWL oraz wykorzystaną biblioteką graficzną.

6.1 Wersja OWL

6.1.1 Lite

- zawiera bazowe elementy OWL i RDF
  - typy: Class, Property, Individual
  - podstawowe nierówności, zależności, charakterystyki
  - elementarna kardynalność
  - adnotacje
- pozwala budować hierarchię elementów
- wymagana jest separacja typów

6.1.2 DL

- zawiera wszystkie elementy języka OWL Lite
- dodatkowo zawiera zaawansowane elementy języka OWL
  - rozwinięta obsługa zależności między elementami podstawowymi
  - kardynalność w pełnej formie
- można go bezpośrednio mapować na logikę opisową SHOIN - jest rozstrzygalny

6.1.3 Full

- zawiera wszytskie elementy OWL DL
- nie wymaga separacji typów
- ma mniejsze ograniczenia od OWL DL
- nie ma w nim gwarancji rozstrzygalności dla wnioskowań

Należy zwrócić uwagę, że specyfikacja OWL jest dobrze zdefiniowana (standard W3C) co sprawia, że zachodzi spójność. Zaimplementowanie wersji bardziej rozwiniętej oznacza, że wymogi dla wersji niższej także zostaną spełnione.

6.2 Proponowane biblioteki do wizualizacji grafów

6.2.1 Prefuse

Prefuse jest elastycznym pakietem dostarczającym programiście narzędzia do przechowywania danych, manipulowania nimi oraz ich interaktywnej wizualizacji. Biblioteka jest rozwijana w całości w języku Java. Może być wykorzystana do budowania niezależnych aplikacji, wizualnych komponentów rozbudowanych aplikacji oraz tworzenia apletów.

Podstawowe cechy i elementy:

- kilkadziesiąt algorytmów i metod wizualizacji danych m.in: ForceDirectedLayout, RadialTreeLayout, NodeLinkTreeLayout, SquarifiedTreeMapLayout
- dynamiczne rozmieszczanie i animacje
- transformacje, przekształcenia geometryczne oraz przybliżanie/oddalanie obrazu
- podstawowym elementem struktury danych jest krotka
- krotki mogą być tworzone bezpośrednio w aplikacji lub na podstawie zewnętrznych danych
- wbudowany język zapytań do filtrowania danych
- tworzenie struktur danych na podstawie zewnętrznych plików (CSV, XML) oraz bazy danych
- klasy wspomagające synchronizację danych pomiędzy tabelami Prefuse a bazą danych
- Prefuse posiada licencję BSD

6.2.2 Piccolo

Piccolo jest zastawem narzędzi używanych przy tworzeniu graficznych aplikacji. Często wykorzystywana do tworzenie interfejsów użytkownika. w których elementy są przybliżane i oddalane. Istnieją trzy wersje tej biblioteki: Piccolo.Java, Piccolo.NET oraz PocketPiccolo.NET. Posiada Licencje BSD.

6.2.3 JUNG (Java Universal Network/Graph Framework)

Biblioteka przeznaczona do wizualizacji danych za pomocą grafów oraz sieci. Umożliwia wizualizację nie tylko grafów prostych, ale m.in. multigrafów, digrafów oraz grafów posiadających wagi i etykiety na wierzchołkach i krawędziach. Biblioteka posiada podstawowe algorytmy grafowe. Została napisana w całości w Javie i wydana na licencji BSD.

6.2.4 JGraph

Napisana w pełni w Javie biblioteka do wizualizacji grafów kompatybilna ze Swingiem. Posiada wiele ciekawych opcji wizualizacji zarówno wierzchołków jak o krawędzi grafów. Poza algorytmami wizualizacji w jej skład wchodzą podstawowe algorytmy grafowe. Została wydana na licencji LGPL.

7 Rekomendowany wariant

OWL

Biblioteka Po uważnym przejrzeniu bibliotek najbardziej użyteczne wydają się Prefuse oraz Piccolo. Ze względu na dostępność dużej ilości przykładowego kodu wykorzystującego Prefuse w portalSubsystem wykorzystana zostanie biblioteka Prefuse. Ponadto opinie wyrażone w pracy magisterskiej Andrzeja Jakowskiego silnie przemawiają na korzyśc Prefuse.

8 Strategia i wstępny harmonogram

Ze względu na doświadczenie zespołu z Rational Unified Process (trzej członkowie zespołu uprzednio zrealizowali projekt w tej metodyce), zostanie on zastosowany z uwzględnieniem stosowanych dla charakteru projektu modyfikacji, w szczególności:

- celem projektu jest wytworzenie biblioteki, więc nie pojawią się typowe diagramy warstwy danych
- model interfejsu graficznego zostanie zastąpiony modelem interfejsów/funkcjonalności zewnętrznych udostępnianych przez pakiety i/lub klasy
- modele dynamiki zostaną okrojone do ilości faktycznie potrzebnej programistom

Pomimo ustalenia harmonogramu z terminami oddania dokumentów należy wziąć pod uwagę charakter metodyki RUP, która zakłada przyrostowe wytwarzanie dokumentacji - w późniejszych etapach projektu pojawią się zmodyfikowane wersje wytorzonych wcześniej dokumentów.

Literatura